Neighbor-List Reduction: Optimization for Computation of Molecular van der Waals and Solvent-Accessible Surface Areas

JÖRG WEISER, ARMIN A. WEISER,* PETER S. SHENKIN, W. CLARK STILL

Department of Chemistry, Columbia University, New York, New York 10027

Received 7 October 1997; accepted 30 December 1997

ABSTRACT: A general, fast, and exact optimization, called neighbor-list reduction (NLR), is presented, which can be used to accelerate the computation of hard-sphere molecular surface areas. NLR allows selected neighbors of a central atom to be removed from the computation in a preprocessing step, thus allowing the calculation of the atom's surface area to proceed with a shorter list of neighbors. The atoms removed are those having intersections with the central atom falling entirely within unions of other atoms' intersections with the central atom. We describe explicit methods for two levels of neighbor-list reduction: 3NLR considers three hard spheres at a time—the central atom, the candidate for removal, and one other neighbor; whereas 4NLR considers two other neighbors. We demonstrate the correctness and efficiency of this optimization by means of a modified version of the NACCESS program, which computes atomic and molecular surface areas numerically. As test cases we used compounds of different size and class, with and without explicit hydrogens. When van der Waals surface (vdWSA) is computed, the NLR methods reduce the length of the neighbor list by as much as 41%; when solvent-accessible surface area (SASA) is computed, the reduction is as great as 74%. The overall speed improvement due to these reductions is a factor of only about 1.2 for vdWSA, but is about 2.0 for

* Present address: Fachbereich Mathematik 3, Technische Universität Berlin, Berlin, Germany

Correspondence to: J. Weiser; e-mail: joerg@still3.chem.columbia.edu

Contract/grant sponsor: Deutsche Forschungsgemeinschaft

© 1998 John Wiley & Sons, Inc.
CCC 0192-8651 / 98 / 070797-12

the computation of SASA, in the context of this particular program. All 39,554 calculated atomic surface areas (vdWSA and SASA) were found to be identical to within 0.001 Å^2 to those obtained without NLR. © 1998 John Wiley & Sons, Inc. J Comput Chem 19: 797–808, 1998

Keywords: van der Waals surface area (vdWSA); solvent-accessible surface area (SASA), numerical calculation; neighbor list; united- and all-atom representations; protein structures

Introduction

A molecule is often represented as a set of overlapping spheres, each characterized by a van der Waals radius. The definition of the solvent-accessible surface area (SASA) was given by Lee and Richards,[1] who presented the image of rolling a sphere, representing a solvent molecule, over the surface of a protein. The solvent-accessible surface is described by the locus of points swept out by the center of the solvent sphere. Much of the current interest in the SASA is due to the observation that, at least for nonpolar molecules, the free energy of aqueous solvation is roughly proportional to this quantity,[2] which is in turn roughly proportional to the number of solvent molecules that can contact the solute molecule. This has made the SASA a central quantity in several implicit solvation models used in molecular mechanics. Many numerical[1,3–10] and analytical[11–23] methods have been developed to calculate the SASA.

Gibson and Scheraga[24] presented a surface area optimization method similar in some ways to the NLR methods described here. They showed how, in the computation of analytical molecular volumes, certain intersections of spheres (including higher order intersections) could be removed from the computation. This speeded the subsequent analytical surface area calculation they described. The NLR methods presented here are, in contrast, applicable to a wide variety of surface area computation algorithms, numerical as well as analytical, including those that do not consider the volume or higher-order intersections in the course of the calculation.

Algorithms used for determining atomic surfaces depend on knowing which other atoms in the system are its neighbors; that is, which are close enough to the central atom (whose area we wish to calculate) to intersect it. We have found that many such neighbors of central atoms are irrelevant: removing them from the neighbor list does not change the exposed surface area of the central atom or the derivatives of this quantity with respect to atomic position. Because many methods for computing these quantities involve iteration over neighbor lists of atoms, elimination of irrelevant neighbors speeds up the area computation without altering the result. We demonstrate the effectiveness and accuracy of the neighbor-list reduction (NLR) optimization using altered versions of the program NACCESS,[25] which numerically computes SASA, or, in the limit of zero solvent radius, van der Waals surface areas (vdWSA).

In the following sections we present the NLR method and compare the timings and results of vdWSA and SASA computations on several compounds of different size (121 to 4550 atoms) and class (proteins, RNA, and an intermolecular complex) both with hydrogens (all-atom representation) and without hydrogens (united-atom representation).

Method

We consider a molecule to consist of a number of atoms, N, with given coordinates for the atomic centers. Each atom has a van der Waals radius; the values used in this work are shown in Table I. The vdWSA is the exposed surface of this system of spheres; the SASA is a similar surface obtained after adding the radius of a water probe (1.4 Å) to the van der Waals radii. We also use these terms to refer to the exposed surface areas of individual atoms.

Neighbor-list reduction relies on the fact that the removal of an atom, j, cannot affect the exposed surface area of another atom, i, which it overlaps, if the i–j intersection is fully included within the intersection of some other atom, k, with i. Nor can its removal have an effect if the i–j intersection lies fully within the union of several other neighbors' intersections with i. In these situations, j can be removed from the neighbor list of

TABLE I.
Atomic van der Waals Radii.

Element	Radius (Å)
H	1.00
C	1.70
N	1.65
O	1.60
S	1.90
P	1.90
Cl	1.80

i without affecting the exposed surface area of i or the total surface of the molecule. In the following, we present conditions for elimination of a j atom due to the presence of a single additional neighbor, k, (3NLR) and due to the presence of two additional neighbors, k and m (4NLR). These methods involve simultaneous consideration of three and four hard spheres, respectively.

The most commonly used NLR-like method is the construction of a proximate neighbor list: all atoms j, that intersect with atom i, are stored on the neighbor list of i; all atoms, which do not intersect with i, are removed. As only two atoms are involved in this process, we refer to this as 2NLR in the following. An additional 2NLR elimination can, in principle, take place when one atom is completely included within another. We perform this check as well; however, this never occurs in our test set, even for all-atom molecular representations. Therefore, when we speak of 2NLR in this study, we are really talking about the construction of a neighbor list that includes all intersecting atom pairs. Because the interatomic distances are needed for 3NLR and 4NLR, we store this neighbor list for subsequent use.

The descriptions of the 3NLR and 4NLR algorithms shown in Schemes 1 and 2 utilize several common data structures. Full3[i] and Full4[i] are neighbor-list arrays that give, for each atom, i, the list of all neighbor atoms, j. Tri2[i] and Tri3[i] are ''triangular'' neighbor lists, containing those j atoms from the corresponding Full arrays such that $j > i$. Tri2[i] is the upper triangular portion of the symmetric distance matrix populated by 2NLR, Full3[i] and Tri3[i] are filled during the 3NLR process. The 4NLR process populates Full4[i], which ends up with the j atoms that survive both reduction processes. Full4[i] contains the list of atoms whose presence must be explicitly considered when the exposed surface of atom i is calculated. Tri2[i], Tri3[i], and Full3[i] are intermediate arrays and may be discarded after Full4[i] is filled. In the following, the symbol d_{AB} denotes the distance between points A and B.

SCHEME 1. 3NLR algorithm. Prior to 3NLR, the atoms in Tri2[i] are sorted in order of increasing distance from I.

```
for( i from 1 to N − 1 ) {
  for( j in Tri2[i] ) {
    for( k in Full3[i] ) {
      if( d_IK > d_IJ or d_JK > d_IJ ) {
        next k
      }
      if( k eliminates j by Equation 1 ) {
        next j
      }
    }
    add j to Tri3[i] and to Full3[i]
    add i to Full3[j]
  }
}
```

3NLR ALGORITHM

In Figure 1, the intersection of sphere i with sphere j lies entirely within the i–k intersection. It should be clear from the diagram that, because of this, if sphere j is removed, the exposed surface of sphere i does not change and vice versa; thus, sphere j may be removed from the neighbor list of sphere i and sphere i may be removed from the neighbor list of sphere j. If d_{IJ} is the largest edge of the IJK triangle, as shown in Figure 1, then the condition for removal of j by k is:

$$\beta \geq \alpha + \gamma \tag{1}$$

where α, β, and γ are as described in Figure 1.

Scheme 1 describes the 3NLR procedure as used in the work described here. This algorithm is not guaranteed to eliminate all spheres which can, in principle, be removed from atom i's pair list. If d_{IJ} is not the largest edge of the IJK triangle, then atom j may still, in some circumstances, be removable by k; however, in these situations, it will not, in general, be safe to eliminate atom i from j's neighbor list as well. The additional tests necessary to find all such situations are time-consuming, and, in any event, at least some of these j atoms are later removed during the 4NLR procedure. Thus, the 3NLR elimination as we carry it out is safe, but not necessarily exhaustive.

SCHEME 2. 4NLR algorithm. Prior to 4NLR, the atoms in Full3[i] are sorted in order of increasing distance from *I*.

```
for( i from 1 to N − 1 ) {
  for( j in Tri3[i] ) {
    for( k in Full3[i] ) {
      if( k = j or k in Elim4[i] ) {
        next k
      }
      if( d_IK > d_IJ ) {
        add j to Full4[i]
        add i to Full4[j]
        next j
      }
      if( k fails k-coverage condition ) {
        next k
      }
      for( m in Full3[i] ) {
        if( m = j or m = k ) {
          next m
        }
        if( m fails m-coverage condition ) {
          next m
        }
        # current k and m satisfy coverage conditions:
        add j to Elim4[i]
        add i to Elim4[j]
        next j
      }
    }
    # no k and m satisfy coverage conditions:
    add j to Full4[i]
    add i to Full4[j]
  }
}
```

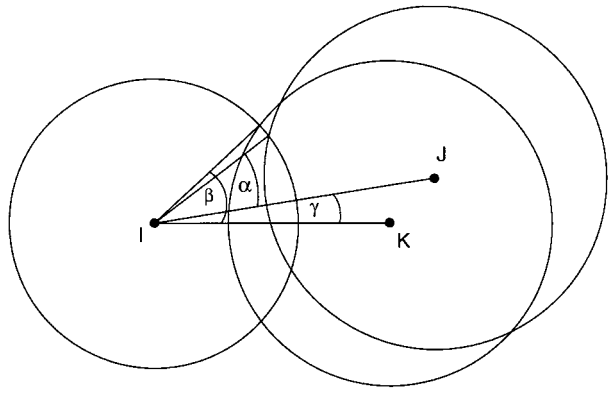

FIGURE 1. 3NLR geometry. Points *I*, *J*, and *K* represent the centers of spheres *i*, *j* and *k*, and lie in the plane of the figure.

4NLR ALGORITHM

Here we look for situations in which the intersection of spheres i and j is included within the union of the i–k and i–m intersections, where j, k, and m are neighbors of i (Fig. 2). If such a situation is found, sphere j can make no contribution to the exposed surface area of i.

In contrast to 3NLR, four hard spheres demand a three-dimensional treatment and simple comparison of angles does not solve the problem of 4NLR. Our solution is based on coverage criteria for the extreme points of the i–j intersection lens. Scheme 2 sets out the 4NLR algorithm. The way the algorithm works is to first look for an atom, k, which includes most of the i–j intersection lens. The criterion that specifies this is termed the k-coverage criterion. If we find such a k atom, we look for another atom, m, which covers the remainder of the i–j intersection. This is tested by what we call the m-coverage criterion. The coverage criteria are given here and the detailed mathematics is in the Appendix.

The k-coverage criterion is met if sphere k includes points B, D, and F (Fig. 2), and if Q and R are both closer to G than to F (Fig. 3). This guarantees that k covers most of the i–j intersection lens, as depicted in the cross-section in Figure 2. All these points are computed from the positions and radii of spheres i, j, and k, as shown in the Appendix.

If k also covers point G, then j is to be removed from the neighbor list of i without checking the m-coverage criterion. This eliminates some j atoms which, as described earlier, were not found in our incomplete 3NLR procedure. This criterion for j elimination is not completely rigorous: if sphere k has a much larger radius than the other spheres, j atoms could be erroneously eliminated. However, even for all-atom molecular systems, such erroneous elimination did not occur in our experience. Therefore, we carry out this elimination in our code.[26]

The 4NLR procedure uses an array called Elim4[i], which is a set of lists, one for each atom, giving, for each atom, i, the atom numbers of neighbors that have already been eliminated from the neighbor list of i. It would seem to make sense that a neighbor of i that has been eliminated should not be used as a k or an m atom in the elimination of further atoms. In fact, we have found, for molecular systems, that it suffices to simply forbid such use as a k atom. Although we do not know if this restriction is sufficient under all conditions of interspherical distances and radii,

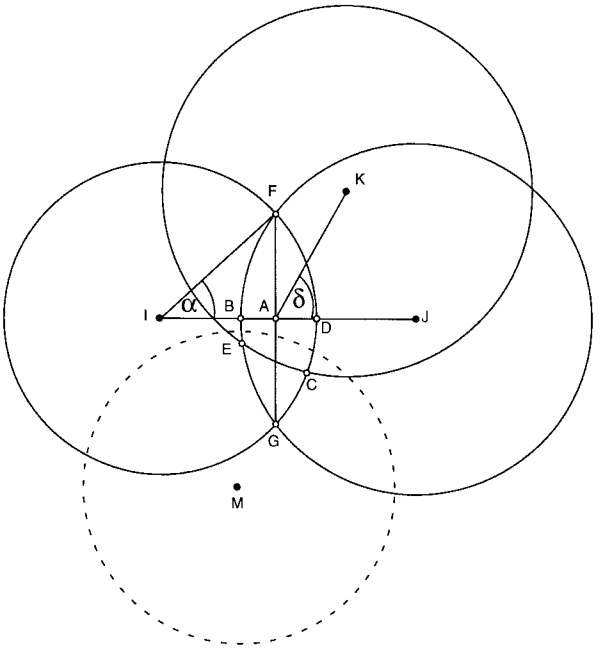

FIGURE 2. 4NLR geometry. Points *I*, *J*, *K*, and *M* represent the centers of the spheres *i*, *j*, *k*, and *m*. Except for *M* all points are in the plane of the figure. Points *A*, *B*, and *D* lie along the line connecting *I* and *J*. *A* is the center of the intersection circle of spheres *I* and *J*; *B* is where the *I*–*J* connecting line intersects the surface of sphere *j* inside sphere *i*; *D* is where this connecting line intersects the surface of sphere *i* inside sphere *j*. *F* and *G* are the points where the *i*–*j* intersection circle touches the *i*–*j*–*k* plane. *E* is the point in the plane where the surface of *k* touches the sphere *j* inside sphere *i*; *C* is where it touches sphere *i* inside sphere *j*.

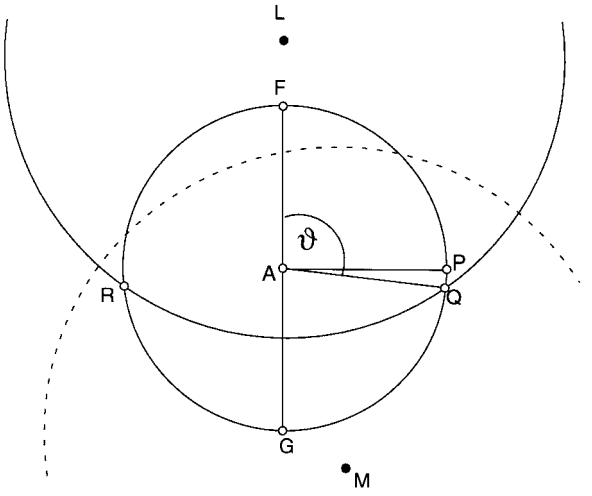

FIGURE 3. The view in this figure is perpendicular to that shown in Figure 2; the *i*–*j* intersection circle lies in the plane of the figure, as do all the labeled points except *M*. We are looking down the *I*–*J* axis. Points *A*, *F*, and *G* are the same points as those labeled in Figure 2. *Q* and *R* are the points where sphere *k* touches the *i*–*j* intersection circle, and point *P* lies on the intersection circle, perpendicular to the line *F*–*A*–*G*. *L* is the projection of *K* to the plane of this figure and is on the line *F*–*A*–*G*. The intersection of *k* with the plane of the figure is shown, centered on *L*.

it appears to suffice for molecular systems, and is embodied in the method we use, as described in Scheme 2.

The criterion ($d_{IK} > d_{IJ}$) in Scheme 2 that bypasses the 4NLR test is not required for correctness; indeed, in all-atom representations, it sometimes results in retention of j atoms which could be eliminated. However, these atoms are few in number, and finding all of them does not justify the extra CPU time that would otherwise be spent searching for them. The picture behind this optimization is that because, in molecular systems, atomic radii do not vary dramatically, it is relatively unlikely that a k atom farther from i than j will cover most of the i–j intersection.

If all atomic radii were equal, the m-coverage criterion would be met if sphere m included points C, E, G, R, and Q, and if $d_{MG} < d_{MF}$. If, on the other hand, sphere m has a radius much greater than i and j, it could cover these points and still not cover the entire surface between C and Q. In Figure 4, the center of m is far to the upper left of the figure, and the straight line CQ represents a portion of the surface of m in the limit of infinite radius. Even though C and Q are on this surface, U is still external to m, by a distance of d_{TU}.

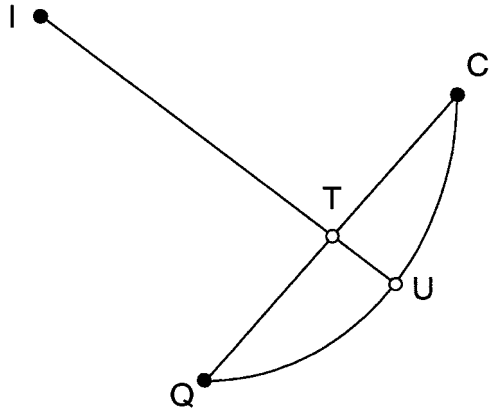

FIGURE 4. *I*, *C*, and *Q* are the same points as in Figures 2 and 3, respectively. The distance between *I* and *U* represents the radius of *i*. *T* is half way on *C*–*Q*. *I*, *T* and *U* are on the same line.

Therefore, we introduce a ''safety factor,'' $\varepsilon_C = d_{TU}$, for point C as well as an analogous value, ε_E, for point E. If the radius of m after subtracting ε_C covers C and its radius after subtracting ε_E covers E, and its radius subtracted by the larger of both ε_C and ε_E covers Q and R, then 4NLR works correctly for all radii. Because the vdW radii used for molecules are never infinite, the introduction of ε_C and ε_E may lead to nonelimination of some atoms, but it is guaranteed to be safe in the sense that if candidate m-atoms are improperly ignored, the net effect is an incomplete optimization, rather than an incorrect result.

For molecules in the all-atom representation, we apply the m-coverage criterion as just described, checking that m includes points G, R, and Q, and, with m's radius decreased by ε_C and ε_E, points C and E, respectively; however, we found no situations in which ε_C and ε_E were necessary in applying the coverage criterion to points Q and R; therefore, we applied the method in this form. For molecules in the united-atom representation, where atomic radii are all very similar, it proved sufficient to simply check that m includes G, R, and Q, and this form of the criterion was used for these systems.

Results and Conclusions

We have calculated the vdWSAs (Table II) and SASAs (Table III) of seven united-atom compounds (five proteins, one RNA, and a model vancomycin complex) and two all-atom compounds (for the largest protein and the RNA). To build the all-atom systems, hydrogens were constructed on the heavy atom backbone with standard bond lengths and angles, using the MacroModel program.[27] All coordinates were obtained from the Brookhaven Protein Data Bank.[28]

To check the validity of our NLR optimizations for distorted structures such as those that occur during free-energy simulations, we carried out a 10-ps, 300 K molecular dynamics (MD) simulation of all-atom crambin and stored a structure once every picosecond. The saved structures were then

TABLE II.
vdWSA Calculations.

Compound[a]	No. of atoms	vdWSA [Å^2]	2NLR pairwise insections (CPU s)	3NLR pairwise insections (CPU s)	4NLR pairwise intersections (CPU s)	CPU time for 2NLR–4NLR [s][b]	NACCESS surface calculation [s]	NACCESS / 2NLR surface calculation [s]	NACCESS / 3NLR surface calculation [s]	NACCESS / 4LNR surface calculation [s]
1van	121	1505	458 (0.001)	433 (95%) (0.001)	415 (91%) (0.002)	0.004	0.12	0.08	0.08	0.07
1crn	327	4252	1209 (0.004)	1151 (95%) (0.002)	1079 (89%) (0.005)	0.012	0.33	0.21	0.20	0.19
2ins	770	9838	2832 (0.019)	2656 (94%) (0.006)	2479 (88%) (0.012)	0.041	0.77	0.47	0.45	0.42
1lz1	1029	13270	3741 (0.033)	3552 (95%) (0.008)	3318 (89%) (0.016)	0.062	1.07	0.64	0.62	0.58
5tra[c]	1822	20085	7689 (0.098)	6837 (89%) (0.018)	5982 (78%) (0.029)	0.157	2.02	1.29	1.19	1.04
1kvd	1988	25843	7103 (0.116)	6733 (95%) (0.016)	6272 (88%) (0.033)	0.176	2.08	1.21	1.17	1.08
3app	2366	30403	8554 (0.167)	8094 (95%) (0.019)	7568 (88%) (0.040)	0.242	2.53	1.46	1.42	1.31
5tra_H[c]	2834	20692	12923 (0.271)	8814 (68%) (0.031)	7882 (61%) (0.058)	0.377	3.68	2.03	1.58	1.37
3app_H	4550	32251	18658 (0.721)	11546 (62%) (0.049)	10981 (59%) (0.086)	0.880	6.49	2.90	2.11	1.93

[a] 1van: vancomycin complex with L-lys-D-ala-D-ala; 1crn: crambin; 2ins: insulin; 1lz1: lysozyme; 5tra: transfer ribonucleic acid (yeast); 1kvd: killer toxin from halotolerant yeast; 3app: penicillopepsin; 5tra_H: all-atom 5tra; 3app_H: all-atom 3app.
[b] Including times in previous three columns plus distance sorting of neighbors between 3NLR and 4NLR.
[c] 5tra includes one atom, ''M7,'' in the pdb file to which we assigned a vdW radius of 1.8Å.

TABLE III.
SASA Calculations.

Compound[a]	No. of atoms	SASA [Å^2]	2NLR pairwise insections (CPU s)	3NLR pairwise insections (CPU s)	4NLR pairwise intersections (CPU s)	CPU time for 2NLR–4NLR [s][b]	NACCESS surface calculation [s]	NACCESS / 2NLR surface calculation [s]	NACCESS / 3NLR surface calculation [s]	NACCESS / 4LNR surface calculation [s]
1van	121	1495	2011 (0.002)	1072 (53%) (0.006)	774 (38%) (0.010)	0.02	0.39	0.32	0.20	0.15
1crn	327	2976	6204 (0.009)	3201 (52%) (0.020)	2242 (36%) (0.028)	0.07	1.22	0.94	0.65	0.44
2ins	770	5741	15072 (0.034)	7916 (53%) (0.055)	5583 (37%) (0.072)	0.19	3.19	2.26	1.59	1.05
1lz1	1029	6740	21268 (0.056)	11044 (52%) (0.081)	7650 (36%) (0.102)	0.28	4.59	3.24	2.23	1.47
5tra[c]	1822	14830	36743 (0.132)	19536 (53%) (0.132)	12787 (35%) (0.176)	0.51	7.64	5.72	3.53	2.45
1kvd	1988	10933	42530 (0.166)	22215 (52%) (0.169)	15301 (36%) (0.210)	0.63	9.33	6.42	4.48	2.92
3app	2366	12723	50529 (0.227)	26502 (52%) (0.201)	18457 (37%) (0.264)	0.79	11.30	7.70	5.38	3.51
5tra_H[c]	2834	14863	73749 (0.401)	33819 (46%) (0.404)	19658 (27%) (0.547)	1.50	15.19	10.76	5.93	3.65
3app_H	4550	12740	136727 (1.105)	61623 (45%) (0.903)	35732 (26%) (1.137)	3.51	32.35	19.52	10.57	6.54

[a] 1van: vancomycin complex with L-lys-D-ala-D-ala; 1crn: crambin; 2ins: insulin; 1lz1: lysozyme; 5tra: transfer ribonucleic acid (yeast); 1kvd: killer toxin from halotolerant yeast; 3app: penicillopepsin; 5tra_H: all-atom 5tra; 3app_H: all-atom 3app.
[b] Including times in previous three columns plus distance sorting of neighbors between 3NLR and 4NLR.
[c] 5tra includes one atom, ''M7,'' in the pdb file to which we assigned a vdW radius of 1.8Å.

subjected to NACCESS SASA calculations with and without NLR. As with the undistorted molecules, we found that every atom in each of these structures had the same SASA with and without the NLR treatment to within the 0.001-Å^2 printout accuracy of the NACCESS program.

For a total of 19,777 atoms, which included all atoms in all molecules in our test set, the surface areas (vdWSA and SASA) determined by the unaltered NACCESS program and by the altered program that included the 2NLR, 3NLR, and 4NLR procedures just described agreed within the 0.001-Å^2 printout accuracy of the NACCESS program. This was found to be true in both the context of entire molecules, as described in Schemes 1 and 2, and also in all situations where we constructed ''molecules'' consisting only of three or four single atoms.

In the following sections we compare the CPU times calculated with the original version of the program NACCESS and with our modified version of the program incorporating our 2NLR, 3NLR, and 4NLR optimizations.

The NACCESS program works by taking thin z-slices through the molecule, calculating the exposed arc length for each atom in each slice, and then integrating the arc lengths over the z-dimension to obtain a total area. The z-slice parameter controls the accuracy and speed of the calculation. We chose a z-slice of 0.01 for accurate results, as recommended by the authors of NACCESS.[25] All calculations were performed on a SGI R10000/194-MHz processor, using Fortran code optimized at the −mips4−O2 level. The use of a thicker z-slice would lessen the improvement in speed conferred by NLR; the use of a thinner z-slice would, of course, increase this improvement.

VAN DER WAALS SURFACES

Table II shows the results of vdWSA calculations. 3NLR reduces the number of neighbors to about 95% (proteins) or 89% (RNA) of the 2NLR

values for united-atom representations of molecules. 4NLR gives further reduction to 88% and 78%, respectively. For all-atom molecular representations, NLR is more effective at shortening the neighbor list: here, 3NLR reduces the number of neighbors to about 62% (proteins) and 68% (RNA) of the original 2NLR neighbor-list size, and 4NLR gives further reduction to about 59% and 61%, respectively.

Despite the extent of these reductions, the greatest improvements in speed in the computation of vdWSA come from 2NLR in united-atom systems; that is, the mere inclusion of a neighbor list, rather than neighbor-list reduction itself, greatly speeds the NACCESS algorithm. For example, consider united-atom 3app. Using 2NLR alone, the surface computation takes 1.63 s, including the 0.17 s needed to carry the 2NLR process (i.e., construct the neighbor list). The speed-up over the unaltered NACCESS program is thus a factor of $2.53/1.63 = 1.55$. Continuing to use the speed of the unaltered program as the basis for comparison, the use of 3NLR together with 2NLR gives a factor of $2.53/1.61 = 1.57$ and using 4NLR as well gives $2.53/1.55 = 1.63$. Thus, although 2NLR, 3NLR, and 4NLR all decrease the overall computation time when their respective overheads are included, the speed improvement is most dramatic for simple 2NLR.

vdWSA calculations of all-atom molecular systems exhibit larger 3NLR speed-ups as well as larger 2NLR speed-ups. For 3app, 2NLR speeds area calculation by a factor of 1.79; adding 3NLR increases the factor to 2.25 and adding 4NLR increases it to 2.31. These factors are again referred to the unaltered NACCESS program.

Even for the smallest systems studied, each of the NLR processes more than pays for itself in CPU time, although the improvements are more dramatic in larger systems. Also, all-atom representations of molecules exhibit greater NLR speed-ups than united-atom systems, presumably because many hydrogens are easily eliminated by the larger heavy atoms.

SOLVENT-ACCESSIBLE SURFACES

Recall that the SASA can be thought of as a surface of an assembly of hard-sphere atoms with augmented vdW radii. Because the increase in the radii does not alter the interatomic distances, each sphere has many more overlapping neighbors in a SASA computation than in a vdWSA computation. The greater overlap of the solvent-augmented atomic spheres should provide a particularly large advantage to neighbor-list reduction methods. Table III shows the SASA results.

Comparison of Tables II and III shows that the unreduced neighbor list grows by a factor of five to seven when the SASA, rather than the vdWSA, is computed. For united-atom systems, 3NLR reduces the SASA neighbor list to about 53% of its 2NLR size, then 4NLR effects a further reduction to about 36%. For all-atom systems, the figures are about 46% and 27%, respectively. These reductions greatly exceed those observed for vdWSA computations, as expected. Furthermore, even though the neighbor list grows by a factor of five to seven, and even though the NLR reduction is greater, on a percentage basis, for SASA than for vdWSA, the time it takes to carry out the NLR reductions grows by only about a factor of three to five. Thus, the extra neighbor-list reduction possible with SASA comes cheaply.

In contrast with the vdWSA optimization results, SASA computations are speeded up more by the higher order than by the lower order NLR processes. United-atom 3app exhibits speed-up factors, based on the unaltered NACCESS program, of 1.43, 1.95, and 2.63 when 2NLR, 3NLR, and 4NLR, respectively, are added to the computation. All-atom 3app exhibits speed-up factors of 1.57, 2.57, and 3.22, respectively. Thus, although the use of a neighbor list (2NLR) makes NACCESS more efficient by about the same amount in vdWSA and SASA computations, the 3NLR and 4NLR reductions are far more effective in decreasing the CPU time of SASA computations.

Discussion

The results shown in Tables II and III demonstrate that, for all classes and sizes of molecules studied, the combination of 3NLR and 4NLR makes the computation of SASA significantly faster, without altering the accuracy of the calculated surface areas. For the computation of vdWSA, the introduction of a neighbor list (2NLR) greatly speeds the program, and the further addition of 3NLR and 4NLR effects only small additional speed-ups in united-atom compounds. However, 3NLR has a big effect in all-atom representations, again without changing the numerical results. These results were obtained over a test set of 19,777 atoms in nine compounds, as well as in a set of distorted structures of all-atom crambin, which were ob-

tained from molecular dynamics. Thus, we regard the correctness of 3NLR and 4NLR to be practically, if not mathematically, demonstrated for both vdWSA and SASA, and the effectiveness of these methods to be clearly demonstrated for the computation of SASA.

For SASA computations on both united-atom and all-atom systems, speed-up factors in the range of 3 are observed over the original NACCESS program, taking into account the overhead of the NLR procedures themselves; compared to NACCESS with only 2NLR, speed-up factors are in the range of 2. Neighbor lists are reduced to about 37% (united-atom) or 27% (all-atom) of their original size. For the computation of vdWSAs, NLR improvements are less dramatic: a factor of about 1.6 is observed over the unaltered NACCESS program, and nearly all of this is attributable to the simple use of a neighbor list; 3NLR and 2NLR neighbor lists are reduced to about 90% (united-atom) or 60% (all-atom) of their original size, but for vdWSA, the cost of doing this is barely repaid by the subsequent speed improvement in the actual surface calculation, whereas 3NLR has a significant effect on all-atom compounds.

Although 4NLR is more complex than 3NLR, the CPU times for the two processes are approximately the same. This is because 4NLR starts with the already reduced neighbor list created by 3NLR. When the atomic radii are large, as in SASA computations, 3NLR performs a large reduction, which speeds the 4NLR process. The results show that 3NLR and 4NLR always save more CPU time than they take to carry out, even for vdWSA calculations and even for the smallest molecules studied, and that the saving due to these processes is large for SASA calculations.

NLR has been shown to work for vdW surface areas (vdWSA) and solvent-accessible surface areas (SASA). The use of the numerically based NACCESS program in this study was intended to demonstrate simply the correctness and efficacy of the NLR optimizations in the context of a single, widely available program. However, NLR is likely to be useful in most methods (both analytical and numerical) of computing hard-sphere molecular surfaces or volumes. For example, we expect that the NLR methods described here would further optimize the methods described by Scheraga et al.,[24,29] if applied as a preprocessing step; however, we have not tested this.

NLR is also applicable to the computation of certain similar or analogous but not identical quantities, such as the Gaussian representation of molecular shape presented by Grant et al.[30] and the solvent-shell volumes utilized by Stouten et al.[31] We expect the use of NLR to speed these methods considerably without degrading their accuracy.

Acknowledgments

The authors thank Simon Hubbard for making his program NACCESS available for the calculations.

Appendix

The following symbols are used:

i, j, k	atom numbers
I, J, K, M	centers of atoms i, j, k, m
r_i	radius of atom i
d_{IK}	distance between points I and K
$\mathbf{B}$	vector from origin to point B
$\mathbf{AP}$	vector from point A to point P
$\mathbf{n}_{JI}$	unit vector from point J to point I
ΔIJK	triangle of points I, J, and K

3NLR

The distance criteria used in Scheme 1 may be described as follows. Comparison is done on the basis of the stored distances from 2NLR:

$$d_{JK} < d_{IJ}, \qquad d_{IK} < d_{IJ} \tag{1}$$

The angles α, β, and γ defined in Figure 1, may be computed using the law of cosines:

$$\cos\alpha = \frac{r_i^2 + d_{IJ}^2 - r_j^2}{2 r_i d_{IJ}} \tag{2}$$

$$\cos\beta = \frac{r_i^2 + d_{IK}^2 - r_k^2}{2 r_i d_{IK}} \tag{3}$$

$$\cos\gamma = \frac{d_{IJ}^2 + d_{IK}^2 - d_{JK}^2}{2 d_{IJ} d_{IK}} \tag{4}$$

Sphere j can be reduced from the neighbor list of atom i if:

$$\beta \geq \alpha + \gamma \tag{5}$$

Because the cosine is a steadily decreasing function over the range 0° to 180°, a more efficient

procedure than eq. (5) is possible:

$$\cos\beta \leq \cos(\alpha + \gamma) \tag{6}$$

For the right-hand side we can use the formula for the cosine of the sum of the two angles:

$$\cos\beta \leq \cos\alpha\cos\gamma - \sin\alpha\sin\gamma \tag{7}$$

Eq. (7) can be written as:

$$\cos\beta \leq \cos\alpha\cos\gamma - \sqrt{(1-\cos^2\alpha)(1-\cos^2\gamma)} \tag{8}$$

The condition expressed by eq. (8) is entirely equivalent to that expressed by eq. (5) if the sum of β and γ is below 180°, which is guaranteed by eq. (1), at least for molecular systems. Eq. (8) is more efficient for evaluating than eq. 5, because three arc-cosines are replaced by a few additions and multiplications and a single square root. Therefore, eq. (8) is used in the code to evaluate the condition for 3NLR.

4NLR

In the following, the details of the 4NLR algorithm and the calculations of the points are presented in algorithmic order (Scheme 2). Given an i and a j atom, we calculate the coordinates of A, B, and D as follows:

$$\mathbf{B} = \mathbf{J} + r_j\mathbf{n}_{JI} \tag{9}$$

$$\mathbf{D} = \mathbf{I} - r_i\mathbf{n}_{JI} \tag{10}$$

$$\cos\alpha = \frac{r_i^2 + d_{IJ}^2 - r_j^2}{2r_i d_{IJ}} \tag{11}$$

$$\mathbf{A} = \mathbf{I} + \mathbf{n}_{IJ}\cdot r_i\cos\alpha \tag{12}$$

To check whether k includes the points B and D, we compare squared distances between K and these points with r_k^2. If the criterion is met, we calculate the position of F as follows:

$$\cos\delta = \frac{d_{AK}^2 + d_{AJ}^2 - d_{JK}^2}{2d_{AK}d_{AJ}} \tag{13}$$

$$\mathbf{F} = \mathbf{A} - \frac{r_i\sin\alpha}{\sin\delta}(\mathbf{n}_{AJ}\cos\delta - \mathbf{n}_{AK}) \tag{14}$$

We now check whether k includes F; if not, k coverage fails. If k includes F, we determine whether j can be excluded without the consideration of m atoms. This is a sort of ''extended 3NLR'' step, and is described in the text. The test requires that k covers G as well as F; however, we can derive an equivalent condition that does not require computation of the position of G. In Figure 3, if sphere k intersects the i–j intersection circle at two points, Q and R, we can write:

$$\cos\vartheta = \frac{d_{AL}^2 + d_{AQ}^2 - d_{LQ}^2}{2d_{AL}d_{AQ}} \tag{15}$$

It is not necessary to calculate point L, because:

$$r_k^2 = d_{LQ}^2 + d_{KL}^2 \tag{16}$$

and:

$$d_{AK}^2 = d_{KL}^2 + d_{AL}^2 \tag{17}$$

Combination of eq. (16) and eq. (17) gives:

$$d_{AL}^2 - d_{LQ}^2 = d_{AK}^2 - r_k^2 \tag{18}$$

As:

$$d_{AL} = d_{AK}\sin\delta \tag{19}$$

and:

$$d_{AQ} = d_{AF} \tag{20}$$

applying eqs. (18), (19), and (20) to eq. (15) leads to:

$$\cos\vartheta = \frac{d_{AF}^2 + d_{AK}^2 - r_k^2}{2d_{AK}d_{AF}\sin\delta} \tag{21}$$

using expressions previously derived. If the i–j intersection circle is completely included within k (or possibly tangent to k in this manner), the right-hand side of eq. (21) will be less than or equal to -1. If this occurs, then j can be eliminated from I's neighbor list, and vice versa. This fulfills the extended 3NLR criterion[26] referred to in our discussion of Scheme 2.

If j is not removed in this way, we need to check whether Q and R are both closer to G than to F. If so, then we will have $-1 < \cos\vartheta < 0$, and the k-coverage criterion is fulfilled. In this event, we need to evaluate m-coverage for m-atom candidates, as described in Scheme 2.

To evaluate m-coverage, we need to compute the positions of G, Q, and R. This is done as follows:

$$\mathbf{G} = \mathbf{A} + \frac{r_i\sin\alpha}{\sin\delta}(\mathbf{n}_{AJ}\cos\delta - \mathbf{n}_{AK}) \tag{22}$$

The positions of Q and R are calculated using $\cos\vartheta$, previously computed, and $\sin\vartheta$, obtained by taking $\sqrt{1-\cos^2\vartheta}$, because ϑ is in the second quadrant (Fig. 3). Then:

$$\mathbf{AP} = \mathbf{n}_{AJ} \times \mathbf{AG} \tag{23}$$

$$\mathbf{Q} = \mathbf{A} + \mathbf{AF}\cos\vartheta + \mathbf{AP}\sin\vartheta \tag{24}$$

$$\mathbf{R} = \mathbf{A} + \mathbf{AF}\cos\vartheta - \mathbf{AP}\sin\vartheta \tag{25}$$

We also need points C and E. To calculate C, imagine that **IG** is translated without rotation so that $G^* = C$. This gives ΔICS with sides u, v, r_i and angles μ, υ, ω (Fig. 6). μ, υ, ω can be computed using α, β, γ (Fig. 5). $\cos\alpha$ was computed in eq. (11). The sides and other angles can be calculated using:

$$\cos\beta = \frac{r_i^2 + d_{IK}^2 - r_k^2}{2 r_i d_{IK}} \tag{26}$$

$$\cos\gamma = \frac{d_{IK}^2 + d_{IJ}^2 - d_{JK}^2}{2 d_{IK} d_{IJ}} \tag{27}$$

$$\mu = \alpha + \beta - \gamma \tag{28}$$

$$\upsilon = \alpha - \beta + \gamma \tag{29}$$

$$\omega = \pi - \mu - \upsilon = \pi - 2\alpha \tag{30}$$

According to the law of sines:

$$u = r_i \frac{\sin\mu}{\sin\omega} \tag{31}$$

$$v = r_i \frac{\sin\upsilon}{\sin\omega} \tag{32}$$

$$\mathbf{IC} = v\mathbf{n}_{IF} + u\mathbf{n}_{IG} \tag{33}$$

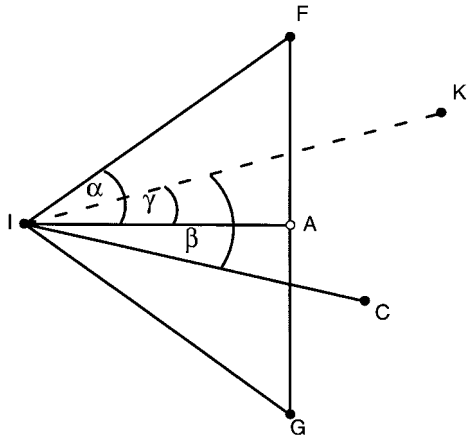

FIGURE 5. The three angles α, β, and γ. All points correspond to the ones shown in Figure 2.

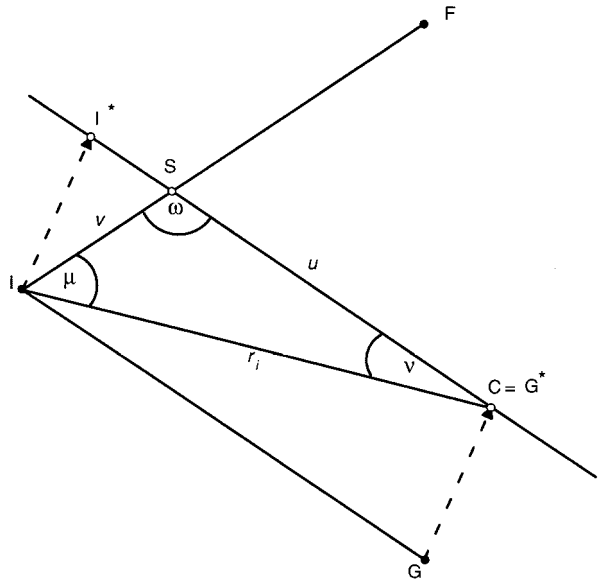

FIGURE 6. The ΔICS. Points I, C, and G correspond to these points in Figure 2. S is the point of intersection of lines I–F and I^*–C.

Applying eq. (31) and eq. (32) to eq. (33) leads to:

$$\mathbf{C} = \mathbf{I} + \frac{\sin(\alpha - \beta + \gamma)}{\sin(2\alpha)}\mathbf{IF} + \frac{\sin(\alpha + \beta - \gamma)}{\sin(2\alpha)}\mathbf{IG} \tag{34}$$

E is calculated analogously:

$$\mathbf{E} = \mathbf{J} + \frac{\sin(\alpha^* - \beta^* + \gamma^*)}{\sin(2\alpha^*)}\mathbf{JF} + \blacksquare\, v\frac{\sin(\alpha^* + \beta^* - \gamma^*)}{\sin(2\alpha^*)}\mathbf{JG} \tag{35}$$

$$\cos\alpha^* = \frac{r_j^2 + d_{IJ}^2 - r_i^2}{2 r_j d_{IJ}} \tag{36}$$

$$\cos\beta^* = \frac{r_j^2 + d_{JK}^2 - r_k^2}{2 r_j d_{JK}} \tag{37}$$

$$\cos\gamma^* = \frac{d_{JK}^2 + d_{IJ}^2 - d_{IK}^2}{2 d_{JK} d_{IJ}} \tag{38}$$

Now that we have calculated points G, Q, R, C, and E, we can apply the m-coverage criterion, once we have computed ε_E and ε_C (see main text):

$$\varepsilon_C = r_i - \left|\frac{\mathbf{C} + \mathbf{Q}}{2} - \mathbf{I}\right| \tag{39}$$

$$\varepsilon_E = r_i - \left|\frac{\mathbf{E} + \mathbf{Q}}{2} - \mathbf{I}\right| \tag{40}$$

We check whether m includes the points G, Q, and R by comparing squared distances, and whether it includes the points C by at least $\varepsilon_C(d_{MC} < r_m - \varepsilon_C)$ and E by at least $\varepsilon_E(d_{ME} < r_m - \varepsilon_E)$. We also require that M be closer to G than to F, using squared distances. If so, we eliminate j and proceed to the next j; if not, the m-coverage criterion is not fulfilled, and we examine the next m, if there is one, for m coverage.

The points C and E do not have to be calculated when applying 4NLR to united atom compounds, because, with the given radii and interatomic distances, C and E are in always, in our experience, included in m if all other conditions for the m coverage are satisfied.

References

1. B. Lee and F. M. Richards, *J. Mol. Biol.*, **55**, 379 (1971).
2. R. B. Hermann, *J. Phys. Chem.*, **76**, 2754 (1972).
3. A. Shrake and J. A. Rupley, *J. Mol. Biol.*, **79**, 351 (1973).
4. T. J. Richmond and F. M. Richards, *J. Mol. Biol.*, **119**, 537 (1978).
5. W. Kabsch and C. Sander, *Biopolymers*, **22**, 2577 (1983).
6. H. Wang and C. Levinthal, *J. Comput. Chem.*, **12**, 868 (1991).
7. S. M. Le Grand and K. M. Merz Jr., *J. Comput. Chem.*, **14**, 349 (1993).
8. R. Abagyan, M. Totrov, and D. Kuznetsov, *J. Comput. Chem.*, **15**, 488 (1994)
9. F. Eisenhaber, P. Lijnzaad, P. Argos, C. Sander, and M. Scharf, *J. Comput. Chem.*, **16**, 273 (1995).
10. (a) A. A. Bliznyuk and J. E. Gready, *J. Comput. Chem.*, **17**, 962 (1996); (b) A. A. Bliznyuk and J. E. Gready, *J. Comput. Chem.*, **17**, 970 (1996).
11. S. J. Wodak and J. Janin, *Proc. Natl. Acad. Sci. USA*, **77**, 1736 (1980).
12. T. J. Richmond, *J. Mol. Biol.*, **178**, 63 (1984).
13. W. Hasel, T. F. Hendrickson, and W. C. Still, *Tetrahed. Comput. Meth.*, **1**, 103 (1988).
14. L. R. Dodd and D. N. Theodorou, *Mol. Phys.*, **72**, 1313 (1991).
15. L. Wesson and D. Eisenberg, *Prot. Sci.*, **1**, 227 (1992).
16. G. Perrot, B. Cheng, K. D. Gibson, J. Vila, K. A. Palmer, A. Nayeem, B. Maigret, and H. A. Scheraga, *J. Comput. Chem.*, **13**, 1 (1992).
17. B. von Freyberg and W. Braun, *J. Comput. Chem.*, **14**, 510 (1993).
18. F. Eisenhaber and P. Argos, *J. Comput. Chem.*, **14**, 1272 (1993).
19. M. Petitjean, *J. Comput. Chem.*, **15**, 507 (1994).
20. D. A. Liotard, G. D. Hawkins, G. C. Lynch, C. J. Cramer, and D. G. Truhlar, *J. Comput. Chem.*, **16**, 422 (1995).
21. S. Sridharan, A. Nicholls, and K. A. Sharp, *J. Comput. Chem.*, **16**, 1038 (1995).
22. M. F. Sanner, A. J. Olson, and J.-C. Spehner, *Biopolymers*, **38**, 305 (1996).
23. R. R. Gabdoulline and R. C. Wade, *J. Mol. Graph.*, **14**, 341 (1996).
24. K. D. Gibson and H. A. Scheraga, *Mol. Phys.*, **62**, 1247 (1987).
25. S. J. Hubbard and J. M. Thornton, NACCESS (Version 2.0) Computer Program, Department of Biochemistry and Molecular Biology, University College London, 1993. The source is available via http://www.biochem.ucl.ac. uk/~roman/naccess/naccess.html.
26. The rigorous algorithm would be as follows: If point C, as calculated in the Appendix, lies within sphere j, then check, whether sphere m, with its radius decreased by ε_G (ε_G is calculated analogoulsy to ε_C, but Q is substituted by G; see Fig. 4), includes G and C. If this is the case, then eliminate j from Full4[i] and eliminate i from Full4[j].
27. F. Mohamadi, N. G. J. Richards, W. C. Guida, R. Liskamp, M. Lipton, C. Caufield, G. Chang, T. Hendrickson, and W. C. Still, *J. Comput. Chem.*, **11**, 440 (1990). (We used MacroModel, version 6.0.)
28. F. C. Bernstein, T. F. Koetzle, G. J. B. Williams, E. F. Meyer Jr., M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi, *J. Mol. Biol.*, **112**, 535 (1977).
29. J. D. Augspurger and H. A. Scheraga, *J. Comput. Chem.*, **17**, 1549 (1996).
30. (a) J. A. Grant and B. T. Pickup, *J. Phys. Chem.*, **99**, 3503 (1995). (b) J. A. Grant, M. A. Gallardo, and B. T. Pickup, *J. Comput. Chem.* **17**, 1653 (1996).
31. P. F. W. Stouten, C. Frömmel, H. Nakamura, and C. Sander, *Mol. Simul.*, **10**, 97 (1993).